NENE 2014 September 8-11 PORTOROŽ SLOVENIA
23rd International Conference Nuclear Energy for New Europe

Automated Motion of the Forerunner Mobile Manipulator Over the Tube Sheets

Tomislav Tomašić
INETEC – Institute for nuclear technology
Dolenica 28
10250 Zagreb, Croatia
tomislav.tomasic@inetec.hr

Andrej Jokić[1], Ante Bakić[2]
[1]Faculty of Mechanical Engineering and Naval Architecture
Ivana Lučića 5, 10000 Zagreb, Croatia
andrej.jokic@fsb.hr
[2]INETEC – Institute for nuclear technology
Dolenica 28, 10250 Zagreb, Croatia
ante.bakic@inetec.hr

ABSTRACT

This paper describes algorithm for the automated motion over the tube sheets of steam generators implemented in the Forerunner manipulator. Forerunner belongs to a group of lightweight mobile manipulators for eddy current testing and repair actions of heat exchangers, primarily vertical steam generators in nuclear power plants of PWR type. Its task is to move along the tube sheet and position different tools at the entrance of the desired pipe. In order to facilitate the manipulator navigation, the high level of autonomy is integrated in the system so that operator only selects a group of tubes to be tested while the algorithms for automated motion decompose the command on sequential series of moves that must be made. Searching for the optimal path among several thousand tubes represents a real challenge from the mathematical point of view. The robustness of the proposed algorithm can be summarized in the following statement: if there is a solution to make the manipulator positioning in a given group of tubes, the algorithm will find it.

1 INTRODUCTION AND PROBLEM DEFINITION

This article will demonstrate the work on development of the automatic movement of the Forerunner manipulator. A detailed view on the manipulator operation and development is given in [1] and [2].

Typical layout of tube sheets for one half of a heat exchanger of a steam generator is presented on Fig. 1, with small blue crosses representing centres of the tubes. The total number of tubes is commonly in the range of several thousands (e.g., around 5000 tubes). Each tube is labelled with a unique label, e.g., an integer in the range from 1 to N, where N is the total number of tubes on the sheet. We introduce the abbreviation $S \coloneqq \{1, \dots, N\}$ for the set of all tubes.

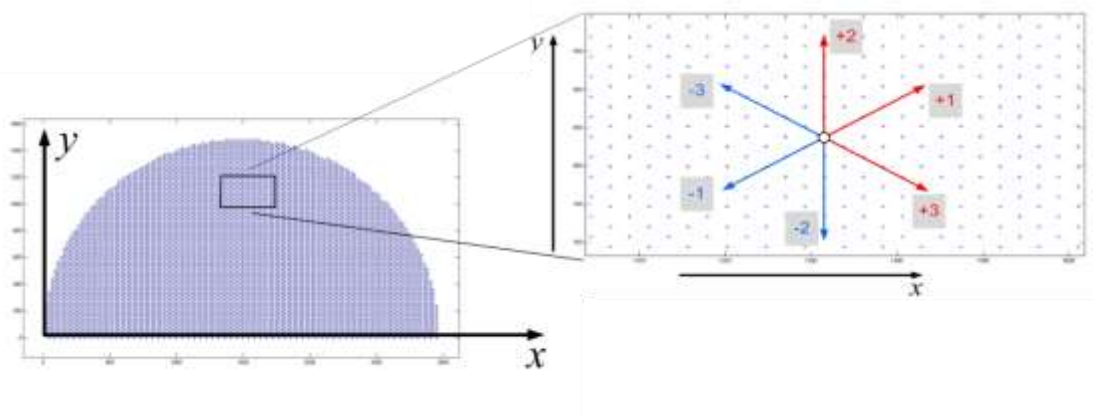

Figure 1. Typical layout of a tube sheet and axis for translational movements of the Forerunner

Tubes on a sheet are arranged in a regular 2D array with three distinguished axes along which the translational movements of the Forerunner are possible. These axis are labelled 1,2 and 3 on Fig. 1. with distinguished positive and negative direction of each axis.

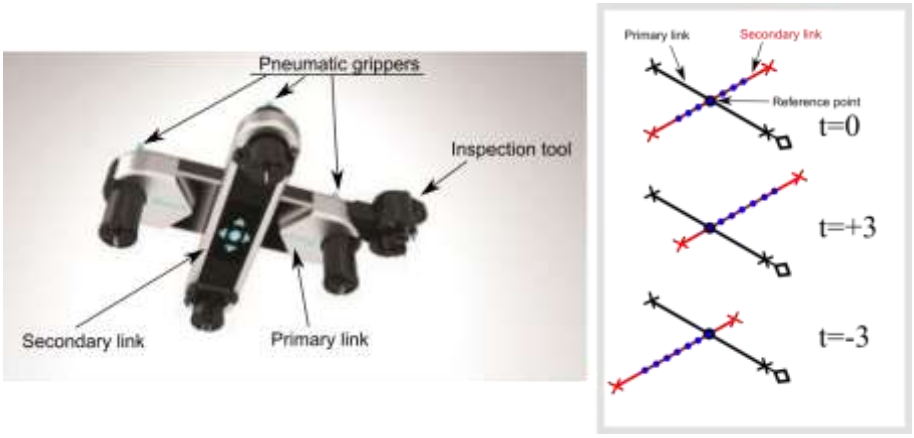

Figure 2. The Forerunner manipulator: definition of links and translational movements

The Forerunner consists of two links: the *primary link* which carries the inspection instruments/tools (see Fig. 2) and the *secondary link* which is used for linear translation movements of Forerunner. The two links can rotate with respect to each other. When the secondary link of Forerunner is aligned with an axis *i* ($i \in \{1,2,3\}$) and is fixed on tubes using a pair of pneumatic grippers, the primary link can translate along the axis of the secondary link (and only along this axis). Alternatively, primary link can be fixed by its own pair of grippers, while the secondary link can translate along its own axis (and not also along the axis of the primary link). The intersection of the primary and secondary link axis is therefore always at the same position with respect to the primary link. This point on the primary link is in the centre of the primary link, i.e., on a half distance between the grippers of the primary link. We will call this point the *reference point* of Forerunner (see Fig. 2.). Primary link has two degrees of freedom: *1*) position *r* of its reference point, where $r \in S$ (the integer value *r* specifies that the reference point of Forerunner is at node with label *r* on the tube sheet); and *2*) orientation $d_p \in \{+1, -1, +2, -2, +3, -3\}$. For example, if $d_p = -2$ then the primary link is aligned parallel to the axis 2, while the link is oriented in such a way that the inspection tool is pointing in negative direction of the axis 2. Let $d_s \in \{1,2,3\}$ denote orientation of the secondary link, i.e., $d_s = i$ means that the secondary link as aligned with the axis i. Note that we do not specify direction of this link as it is symmetrical. For each given orientation d_p we have two possible orientations of the secondary link. For example, if d_p is either +2 or -2, then d_s has to be 1 or 3. The ordered pair (d_p, d_s) defines the configuration of Forerunner.

Note that there are in total 12 possible configurations. Furthermore, with each configuration there are 7 possible relative positions of the secondary link with respect to the reference point of Forerunner, which are indicated on Fig. 2. On the same figure, definition of the variable $t \in \{-3, -2, -1, 0, 1, 2, 3\}$ is presented which is introduced to capture this additional degree of freedom. Therefore, for some given reference point, there are in general 84 possible configurations (12 x 7). Finally, we define the state of Forerunner with an ordered quadruple $s \coloneqq (r, d_p, d_s, t)$.

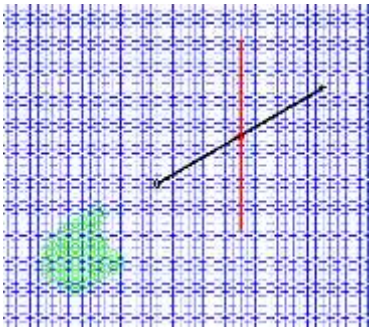

Figure 3. The inspection clique, which is presented with the set of green encircled tubes

With each feasible state s we define the inspection clique $c(s) \subset S$ as the set of tubes that can be inspected if Forerunner is in the considered state s. The shape of the inspection clique, as well as its position relative to the primary link, is presented on Fig. 3 where the clique consists of the green encircled nodes. Black line presents the primary link, with its grippers on the ends of the line. Note that the $c(s)$ in fact depends only on r and d_p as the inspection tool is mounted on the primary link. Finally, note that we can have infeasible states, that is, there are states in which Forerunner cannot come. There are several possible reasons for infeasibility. Some of the tubes can be closed (tube entry is welded) as they are put out of function, what implies that the Forerunner cannot use its grippers on such tubes. Furthermore, some states cannot be reached as Forerunner would with some of its parts collide with the sphere-shaped wall that surrounds the tube sheet. In the remainder, we will use Ω to denote the set of feasible states, that is, if $s \in \Omega$ than this state can be reached.

1.1 The tubes inspection problem

Let $\Gamma \subseteq S$ be a given subset of the set S of all tubes. The set Γ denotes the set of tubes in which each tube has to be inspected using sensors/tools mounted on the Forerunner. Note that it is possible that $\Gamma = S$ (all tubes are scheduled for inspection), while in general $\Gamma \subset \Omega$. Tubes that are closed must not be in the set Γ. Furthermore, let the feasibility set Ω be given. The tubes inspection problem considered in this paper is formulated in terms of the following optimization problem

$$\min_{K,\{s_1,\dots,s_K\}} W(\{s_1, \dots, s_K\}) \tag{1.1}$$

subject to

$$\Gamma \subseteq \bigcup_{i=1}^{K} c(s_i) \tag{1.2}$$

$$s_1, \dots, s_K \in \Omega \tag{1.3}$$

where for the *ordered sequence* of states $\{s_1, \dots, s_K\}$ the objective function W from (1.1) is given by

$$W(\{s_1, \dots, s_K\}) \coloneqq w(s_1, s_2) + w(s_2, s_3) + \cdots + w(s_{i-1}, s_i) + w(s_i, s_{i+1}) + \cdots$$

$$\ldots + w(s_{K-1}, s_K). \tag{2}$$

Function $w(s_i, s_j)$ in (2) is the distance (*shortest path*) between the states s_i and s_j. Note that this distance is in general not equal to the Euclidian distance between the corresponding two reference points r_i and r_j, but is a suitable measure of a minimum number of movements that Forerunner has to do to come from s_i and s_j. More details on the distance and shortest path calculation are given in Section 3.1 below in the text.

Interpretation of the optimization problem (1) is as follows. For a given set Γ we are searching for an *ordered* sequence of states $\{s_1, \ldots, s_K\}$ from which all the tubes in Γ can be inspected. This condition is formulated in the constraint (1.2). Note that in addition to the sequence $\{s_1, \ldots, s_K\}$ itself, the number K of states from which the inspections will be performed is also not *a priori* defined, but is in fact a decision variable. The constraints (1.3) imply that only feasible states are to be considered. Finally, the objective function in (1.1) quantifies the length of the total path Forerunner has to cross while visiting each state once. The order of states in the sequence specifies in which order the states are visited.

Proposition 1. The optimization problem (1) belongs to the class of NP-complete problems.

Proof. Let the set of states $D \coloneqq \{s_1, \ldots, s_K\}$ that satisfy the constraints (1.2) and (1.3) is given. Suppose that the set D is such that for each $s_i \in D$ there is at least one tube in Γ which can be inspected only from the state s_i and from no other state in D. Now, consider the following optimization problem: calculate the optimal ordering of the states in D to minimize objective function in (1.1). Let us call this problem the *ordering problem*. The ordering problem can be reduced to the problem (1) by setting $c(s) = \emptyset$ for all states that are not in the set D and therefore the complexity of problem (1) is at least the same as the ordering problem. To see this, assume that the problem (1) admits polynomial time algorithmic solution. Then the ordering problem could also, by reduction to problem (1), be solvable in a polynomial time. Since the ordering problem is equivalent to the traveling salesman problem [3], and therefore belongs to the NP-complete class (no polynomial time algorithm has yet been found), our assumption that the problem (1) has a polynomial time solution is necessarily wrong. Finally, we conclude that the problem (1) belongs to the class of NP-complete problems. ■

2 ALGORITHMIC SOLUTION

Since the number of feasible states on a tube sheet readily reaches the order of thousands, Proposition 1 implies that we have to abandon attempts to find an algorithm which will solve the problem (1) optimally in any reasonable time. We aim on finding an efficient algorithm for calculating a suboptimal solution to (1) with calculation time in the order of tens of seconds. For that purpose, consider the following optimization problem, which is closely related to the problem (1):

$$\min_{M, \{L_1, \ldots, L_M\}, \sigma(\{L_1, \ldots, L_M\})} W(\sigma(\{L_1, \ldots, L_M\})) \tag{3.1}$$

subject to

$$\Gamma \subseteq \bigcup_{i=1}^{M} \bigcup_{j=1}^{|L_i|} c(L_i[j]) \tag{3.2}$$

$$L_i[\mathrm{j}] \in \Omega \quad \text{for all } j = 1, \ldots, |L_i| \text{ and } i = 1, \ldots, M \tag{3.3}$$

$$L_i \in Z \tag{3.4}$$

where each L_i is a priori known *ordered* set of states and where $|L_i|$ and $L_i[j]$ denote respectively the number of elements in L_i and the j-th element of L_i (note that $L_i[j]$ is a state,

and therefore the constraints (3.2) and (3.3) are well defined). Constraint (3.2) states that the set Γ is contained in the union of inspection cliques from the states of the selected sets $\{L_1, \dots, L_M\}$. In (3.1) the term $\sigma(\{L_1, \dots, L_M\})$ denotes a permutation $\{\tilde{L}_1, \dots, \tilde{L}_M\}$ of the set $\{L_1, \dots, L_M\}$, while the objective function W is defined as follows

$$W(\{\tilde{L}_1, \dots, \tilde{L}_M\}) \coloneqq w(s^*, \tilde{L}_1[e_1^A]) + w(\tilde{L}_1[e_1^B], \tilde{L}_2[e_2^A]) + w(\tilde{L}_2[e_2^B], \tilde{L}_3[e_3^A]) + \dots + w(\tilde{L}_{i-1}[e_{i-1}^B], \tilde{L}_i[e_i^A]) + w(\tilde{L}_i[e_i^B], \tilde{L}_{i+1}[e_{i+1}^A]) + \cdots + w(\tilde{L}_{M-1}[e_{M-1}^B], \tilde{L}_M[e_M^A]) + w(\tilde{L}_M[e_M^B], s^*). \quad (4)$$

where $w(s_i, s_j)$ is the distance (shortest path) between the states s_i and s_j, as in (2). For a set $\tilde{L}_i$, the variable e_i^A is either 1 or $|\tilde{L}_i|$, that is, the state $\tilde{L}_i[e_1^A]$ is either the first or the last element of the ordered set $\tilde{L}_i$. Furthermore, if $e_i^A = 1$ then $e_i^B = |\tilde{L}_i|$, and if $e_i^A = |\tilde{L}_i|$ then $e_i^B = 1$. Although it is not explicitly stated in (3.1), each variable $e_i^A, i = 1, \dots, M$ is a decision variable in (3) as well. Furthermore, note the number M of sets L_i, selected from for a known set Z (see (3.4)), is also not *a priori* defined, but is in fact a decision variable.

The main idea behind the optimization problem (3) is as follows. Note that in problem (1) we are selecting *single states* from the very large set of all feasible states, to obtain the optimal inspection path. Instead, in optimization problem (3) we select *sets of states* L_i from a much smaller number of given sets L_i, and suitably combine the selected L_i's to obtain the solution. The number M of the selected sets L_i will in practice be in order of tens, so the ordering problem (equivalent to the traveling salesmen problem) that is implicitly contained in (3) can be efficiently solved. Each set L_i is a priory defined *ordered* set of states, stored in a data base (set Z from (3.4.)). The order of the states in a set L_i specifies in which order the states from within L_i should be visited by Forerunner. Forerunner "enters" the set L_i either at the first state in the set ($L_i[1]$) or the last state in the set ($L_i[|L_i|]$). If it enters at $L_i[1]$, Forerunner has to follow the ordered sequence of states in L_i and "exists" at $L_i[|L_i|]$. Alternatively, if Forerunner enters the set at $L_i[|L_i|]$, it has to follow the reverse sequence of the remaining states in L_i, and finally exists the set at the state $L_i[1]$. The inspection path, i.e. the ordering of states, within L_i is calculated separately for each L_i in Z, so that it is efficient in the sense of being the shortest path for visiting all the states in that particular L_i. In addition to this, the set Z and its elements L_i have to be such that the following holds: if there exists a solution for inspection path for some given Γ, the optimization problem (3) is necessarily feasible, i.e., solving problem (3) will find a solution. More details of how to construct sets L_i so that the above properties are satisfied are presented in Section 3.2.

The state s^* in (4) can be either a priory determined state which is then the beginning and the end of the inspection path, or this state can also be treated as decision variable in the optimization problem (3).

Suboptimal, but practically computationally efficient, solution to the optimization problem (3) is presented in the following subsections. In particular, the problem (3) is decomposed into two subproblems: *i*) finding suboptimal (in terms of the value of optimization function in (3)) sets $\{L_1, \dots, L_M\}$ from a suitably constructed data base Z, so that the constraints (3.2) and (3.3) are satisfied; and *ii*) finding the optimal order $\sigma(\{L_1, \dots, L_M\})$. Solution to the subproblem (*i*) is presented in Subsection 3.2, where construction of the sets L_i is also presented. Solution to the subproblem (*ii*) is addressed in Subsection (3.3). In section (3.1) we present solution for calculating the function $w(\cdot,\cdot)$ from (4).

2.1 Shortest path

The length of the shortest path between two arbitrary states of Forerunner defines the function $w(\cdot,\cdot)$ from (2) and (4). To precisely specify the problem, we have to assign certain weights (penalties) on the elementary movements of Forerunner. These elementary movements are: rotation of the primary link while the secondary axis is fixed; rotation of the secondary link while the primary link is fixed; and translation of either primary or secondary link to obtain some change Δt in the value t (see Fig. 2). Each more complex path consists of a set of these elementary movements. By setting the weights on the elementary movements, which are positive real numbers, we define the ``artificial'' length of each of these movements. The weights are treated as tuning parameters which can be changed to influence the outcome of the shortest path search algorithm.

The shortest path problem is mathematically formalized as finding the shortest path on an undirected weighted graph – we will call this graph the *state graph* - in which each *node* corresponds to one state s. *Edges* in the state graph connect two states s_i and s_j ($s_i \neq s_j$) if the Forerunner can come from s_i to s_j by a single elementary movement, e.g., by one rotation of either primary or secondary link, or by a single translation for Δt, where $|\Delta t| \leq 7$. Each elementary movement is performed with fixed position of either primary or secondary link. With each edge in the graph, we define its weight correspond to the weight of the corresponding elementary movement. The states that are infeasible can either be removed from the state graph or the weights on all edges connected to these states can be set to infinity.

With the above formulations, shortest path between two arbitrary states can be found in a polynomial time. For this purpose we use Dijkstra's algorithm, see e.g. [3] for detailed presentation and computational complexity quantifications.

2.2 Inspection positions

The set $\boldsymbol{Z}$ and its elements L_i are carefully created to satisfy the following conditions: A) the path of visiting states within each L_i is well defined and is efficient (shortest) in terms of the number of elementary movements of Forerunner; B) there is a *continuity* of inspected tubes when the Forerunner moves along the path within L_i, that is, union of inspection cliques of the states within L_i does not contain any "islands" of uninspected tubes (the "islands of tubes" for which Forerunner would need to "come back" to inspect them once after it has exited the set L_i); C) if a tube can be inspected from some feasible state, this state has to belong to some L_i in Z.

To satisfy properties (A) and (C), each set L_i is composed of sets of states organized in rows (see Fig. 4.). From one state in a row to any other state in the same row, Forerunner can efficiently come using only translations. All the states in some set L_i are characterized with the same configuration (d_p, d_s) of Forerunner; see Fig. 5 for example of a set in the configuration (+2,1). Distance between the states in one row is such that the inspection continuity property (B) is achieved (for illustration see left part in Fig. 4.). Similarly, distance between the rows is selected to preserve this continuity (see right part in Fig. 4.)

To satisfy property (C), for *each configuration* (d_p, d_s) we define in total 63 sets L_i, where the relation between any of the two sets from these 63 sets is that the elements of one set can be obtained from the other set by translation of Δr_p tubes in direction d_p and translation of Δr_s tubes in the direction d_s. Here $\Delta r_p \in \{0, \dots, 6\}$ and $\Delta r_s \in \{0, \dots, 8\}$, and therefore there are 63 sets per configuration ($|\Delta r_p| \cdot |\Delta r_s| = 7 \cdot 9$). If one would choose $\Delta r_p \geq 7$ and/or $\Delta r_s \geq 9$, the elements of the sets L_i would start to overlap. See Fig. 5. for example of

two different sets L_i in configuration (+2,1) where $\Delta r_p = 3$ and $\Delta r_s = 2$. Since there are in total 12 configurations defined by (d_p, d_s), see Section 2.1 for details, the set Z consists of $12 \cdot 63 = 756$ sets L_i. It can be shown that with such construction of Z and its elements the property (C) is necessarily satisfied.

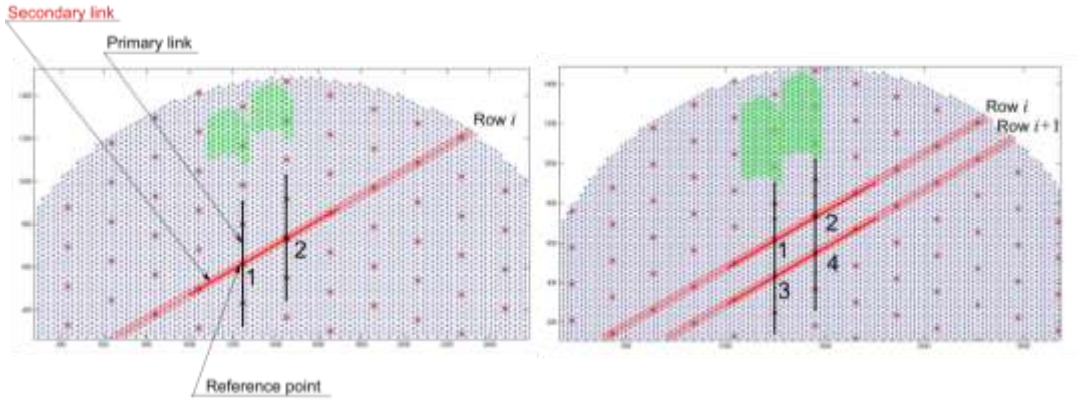

Figure 4. Creation of a set L_i: definition of rows and continuity of inspection

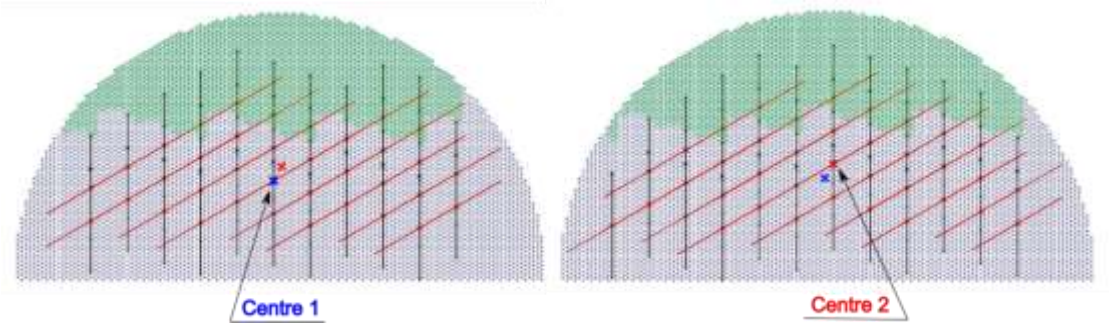

Figure 5. Examples of two sets L_i in the same configuration

Fig. 6. presents an example of a set L_i (the same set as in Fig. 5, left) with distinguished entrance and exit states for an inspection path within L_i: points A and B in the figure. The optimal inspection path within the L_i (example of which is presented by arrows in Fig. 6.) can be calculated using analogous algorithm to the one from Section 3.3 below (the typical number of states in the set L_i is sufficiently small that such computation can be efficiently performed), or can be induced exploiting structural properties of the set L_i. Example of the latter approach goes along the following lines: visit all the states in a single row (black arrows in Fig. 6.) using only simple translations, before moving to the next row (red arrows). Rotations are used only during transitions from one row to the other, while the total number of elementary movements on the path between A and B is minimized.

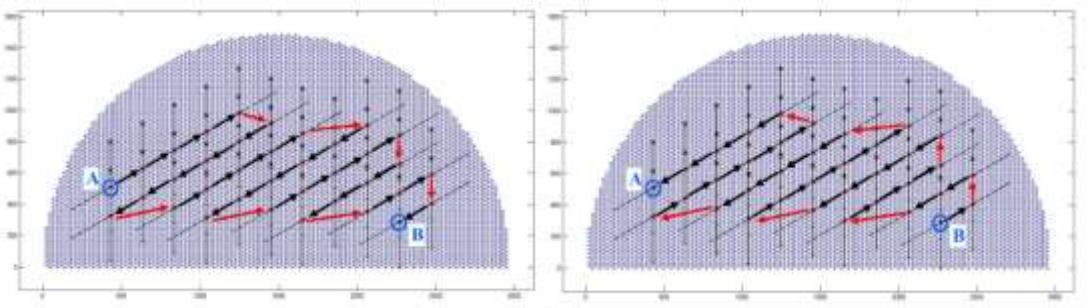

Figure 6. Example of the optimal inspection path within the set L_i

Finally, with the set Z fully defined and for given sets Γ and Ω, finding sets $\{L_1, \dots, L_M\} \subseteq Z$ as a part of solution to problem (3), can be performed using suboptimal greedy algorithm as follows: (***step 1***) from current set Z select L_i from which the largest number of tubes in the current set Γ can be inspected. Take care that infeasible states are removed from the selected L_i. (***step 2***) Remove the selected L_i from the set Z; remove from Γ the tubes that are inspected from the selected L_i; and go to step 1. ***Stopping criteria***: the above two steps are iterated until either Γ becomes empty or the step (1) fails to find a set L_i so that at least one tube from the current Γ can be inspected.

2.3 Inspection path

For a given set $\{L_1, \dots, L_M\}$, the optimal sequence $\sigma(\{L_1, \dots, L_M\})$ of visiting these sets can be formulated in terms of an integer linear programming problem (ILP). For M being in the order of tens, what turns out to be the case in practice, the considered ILP is efficiently solved using e.g. branch-and-bound algorithm [4] based on LP relaxations. Instructions of how to formulate the corresponding ILP are given as follows.

Suppose that s^* from (4) is given. With the abbreviations $e_i := |L_i|,\ i = 1, \dots, M$, consider the graph with the set of nodes $\{s^*, L_1[1], L_1[e_1], \dots, L_M[1], L_M[e_M]\}$ and with the weighted *undirected* edges defined as follows: there is an edge between s^* and each $L_i[1]$ with the weight $w(s^*, L_i[1])$, where $w(\cdot,\cdot)$ is the distance function from Section 3.1; there is edge between s^* and each $L_i[e_i]$ with the weight $w(s^*, L_i[e_i])$; there is an edge between all possible pairs $\big(L_i[1], L_j[1]\big)$ for $i \neq j$, with the corresponding weight $w(L_i[1], L_j[1])$; there is an edge between all possible pairs $\big(L_i[1], L_j[e_j]\big)$ for $i \neq j$, with the corresponding weight $w(L_i[1], L_j[e_j])$; there is an edge between $L_i[1]$ and $L_i[e_i]$ for all $i \in \{1, \dots, M\}$, with the weight of each edge set to ε, where ε is sufficiently small positive real number. Formulating the corresponding ILP follows the lines of formulating the traveling salesman problem on the above described graph, see e.g. [4] and the references therein for details. In particular note that with setting the weight on an edge between entrance and exit states for an inspection path within L_i to some small number, i.e. by setting $w(L_i[1], L_i[e_i]) = \varepsilon$, we ensure that each such edge will be selected in the optimal path. In real-life implementation, each such edge is replaced with the internal L_i path, whose definition has been presented in Section 3.2.

3 CONCLUSIONS

In this paper we have developed and described the algorithm for automated motion planning of Forerunner manipulator while inspecting tube sheets of a steam generator. The algorithm can efficiently solve real-life tube inspection problems characterized with several thousands of tubes. Furthermore, the algorithm is proven to find a solution (inspection path) if a solution theoretically exist.

REFERENCES

[1] F.Jarnjak, “Machine Vision Secondary Positioning System for a VVER Steam Generator Inspection Manipulator”, American Nuclear Society 2009 Annual Meeting, Atlanta, Georgia, USA, June, 2009.

[2] S. Galošić, M. Vavrouš, M. Budimir, “Tube Sheet Runner”, Nuclear Energy For New Europe 2011, Bovec, Slovenia

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, The MIT Press, 2009

[4] C.H. Papadimitriou, K. Steiglitz, Combinatorial optimization: algorithms and complexity; Dover Books on Computer Science, 1998